**SmartProposal Engine MVP代码设计及实施方案**

**一、项目整体结构与规范**

**1.1 项目目录结构**

smart\_proposal\_engine/

├── app.py # 主应用入口

├── requirements.txt # 项目依赖

├── README.md # 项目说明文档

├── .env.example # 环境变量示例

├── app\_config.ini # 应用配置文件

│

├── pages/ # Streamlit多页面应用

│ ├── 1\_📄\_Input\_Processing.py # 内容输入页面

│ ├── 2\_🔍\_Deep\_Analysis.py # 深度分析页面

│ ├── 3\_📋\_Proposal\_Generation.py # 方案生成页面

│ └── 4\_🚀\_One\_Click\_Generation.py # 一键生成页面

│

├── services/ # 业务服务层

│ ├── \_\_init\_\_.py

│ ├── base\_service.py # 服务基类

│ ├── transcription\_service.py # 音频转录服务（基于现有）

│ ├── document\_service.py # 文档处理服务

│ ├── analysis\_service.py # 深度分析服务（基于现有）

│ └── proposal\_service.py # 方案生成服务

│

├── core/ # 核心组件层

│ ├── \_\_init\_\_.py

│ ├── prompt\_manager.py # Prompt模板管理器

│ ├── model\_interface.py # 统一的模型接口

│ ├── document\_processor.py # 文档处理器

│ └── session\_manager.py # 会话状态管理器

│

├── prompts/ # Prompt模板目录

│ ├── transcription/

│ │ └── optimization.md # 转录优化模板

│ ├── analysis/

│ │ ├── customer\_interview.md # 客户访谈分析模板

│ │ ├── business\_negotiation.md # 商务谈判分析模板

│ │ └── internal\_meeting.md # 内部会议分析模板

│ └── proposal/

│ ├── project\_proposal.md # 项目建议书模板

│ ├── quotation\_proposal.md # 商务报价模板

│ └── solution\_brief.md # 解决方案简报模板

│

├── utils/ # 工具函数

│ ├── \_\_init\_\_.py

│ ├── file\_utils.py # 文件处理工具

│ ├── format\_utils.py # 格式化工具

│ └── validation\_utils.py # 验证工具

│

├── assets/ # 静态资源

│ ├── styles/

│ │ └── custom.css # 自定义样式

│ └── images/

│ └── logo.png # 项目Logo

│

├── temp/ # 临时文件目录

├── output/ # 输出文件目录

└── tests/ # 测试目录（MVP暂不实现）

**1.2 统一编码规范**

**文件头部规范**

每个Python文件必须包含以下标准头部：

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

"""

文件路径: smart\_proposal\_engine/services/transcription\_service.py

功能说明: 音频转录服务模块，负责处理音频文件的转录和优化

作者: SmartProposal Team

创建日期: 2025-06-27

最后修改: 2025-06-27

版本: 1.0.0

"""

**类和函数规范**

class ServiceName:

"""

服务类说明

主要功能:

- 功能点1

- 功能点2

使用示例:

service = ServiceName()

result = service.process(data)

"""

def method\_name(self, param1: str, param2: dict) -> ProcessingResult:

"""

方法功能说明

Args:

param1: 参数1说明

param2: 参数2说明

Returns:

ProcessingResult: 返回结果说明

Raises:

ValueError: 异常情况说明

"""

**命名规范**

* 类名：使用PascalCase（如ServiceName）
* 函数和方法名：使用snake\_case（如process\_input）
* 常量：使用UPPER\_CASE（如MAX\_FILE\_SIZE）
* 私有方法：使用单下划线前缀（如\_validate\_input）

**导入规范**

# 标准库导入

import os

import sys

from typing import Dict, List, Optional, Union

# 第三方库导入

import streamlit as st

import google.generativeai as genai

# 项目内部导入

from core.prompt\_manager import PromptManager

from services.base\_service import BaseService

**二、核心文件设计详细说明**

**2.1 主应用入口（app.py）**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

"""

文件路径: smart\_proposal\_engine/app.py

功能说明: SmartProposal Engine主应用入口

"""

# 主要功能：

# 1. 初始化Streamlit应用配置

# 2. 设置页面导航

# 3. 加载全局配置

# 4. 初始化session state

# 5. 应用自定义样式

**2.2 服务层文件设计**

**base\_service.py**

"""

文件路径: smart\_proposal\_engine/services/base\_service.py

功能说明: 所有服务的基类，定义统一接口

"""

# 核心接口：

# - process(): 主处理方法

# - validate\_input(): 输入验证

# - get\_available\_templates(): 获取可用模板

# - configure(): 服务配置

# - format\_result(): 结果格式化

# 统一数据结构：

@dataclass

class ProcessingResult:

content: str

metadata: Dict

source\_type: str

processing\_time: float

model\_used: str

tokens\_consumed: Dict

error: Optional[str] = None

**transcription\_service.py**

"""

文件路径: smart\_proposal\_engine/services/transcription\_service.py

功能说明: 音频转录服务，继承并扩展现有TranscriptionService

"""

# 复用现有代码：

# - 保留现有的TranscriptionService核心功能

# - 适配新的BaseService接口

# - 添加对新输入格式的支持预留

# 主要扩展：

# 1. 统一的process()方法

# 2. 支持Web文本输入的处理

# 3. 与PromptManager集成

**document\_service.py**

"""

文件路径: smart\_proposal\_engine/services/document\_service.py

功能说明: 文档处理服务，支持多种文档格式

"""

# 主要功能：

# 1. 支持DOCX文档解析（使用python-docx）

# 2. 支持PDF文档解析（使用PyPDF2）

# 3. 支持TXT文档读取

# 4. 统一输出为标准文本格式

# 5. 保留文档元数据（页数、创建时间等）

# 预留接口：

# - process\_image(): 图片OCR处理（未来扩展）

# - process\_excel(): Excel文件处理（未来扩展）

**analysis\_service.py**

"""

文件路径: smart\_proposal\_engine/services/analysis\_service.py

功能说明: 深度分析服务，基于现有DeepAnalysisService扩展

"""

# 复用现有代码：

# - 保留DeepAnalysisService的核心分析逻辑

# - 保持现有的analyze\_transcript方法

# 主要扩展：

# 1. 支持多种分析模板切换

# 2. 支持自定义Prompt上传

# 3. 统一的结果格式化

# 4. 与PromptManager深度集成

**proposal\_service.py**

"""

文件路径: smart\_proposal\_engine/services/proposal\_service.py

功能说明: 方案生成服务，生成各类商业文档

"""

# 主要功能：

# 1. 基于分析结果生成方案

# 2. 支持多种方案模板

# 3. 整合企业能力文档（可选）

# 4. 生成专业格式的输出文档

# 核心方法：

# - generate\_proposal(): 生成方案主方法

# - merge\_capability\_docs(): 整合能力文档

# - format\_proposal(): 格式化方案输出

**2.3 核心组件层设计**

**prompt\_manager.py**

"""

文件路径: smart\_proposal\_engine/core/prompt\_manager.py

功能说明: Prompt模板管理器，负责模板的加载、缓存和管理

"""

# 主要功能：

# 1. 从文件系统加载内置模板

# 2. 支持运行时加载自定义模板

# 3. 模板参数化和变量替换

# 4. 模板缓存管理

# 5. 模板版本控制

# 核心方法：

# - load\_template(): 加载模板

# - get\_template(): 获取模板内容

# - register\_custom\_template(): 注册自定义模板

# - list\_available\_templates(): 列出可用模板

**model\_interface.py**

"""

文件路径: smart\_proposal\_engine/core/model\_interface.py

功能说明: 统一的AI模型接口，封装Gemini API调用

"""

# 主要功能：

# 1. 统一的模型调用接口

# 2. 自动重试和错误处理

# 3. Token使用量统计

# 4. 费用计算

# 5. 模型切换支持

# 复用现有配置：

# - 使用现有的models.conf配置

# - 保持现有的API调用模式

**session\_manager.py**

"""

文件路径: smart\_proposal\_engine/core/session\_manager.py

功能说明: 会话状态管理器，管理跨页面的数据流转

"""

# 主要功能：

# 1. 管理Streamlit session state

# 2. 数据在各环节间的传递

# 3. 临时文件管理

# 4. 结果缓存

# 核心方法：

# - save\_result(): 保存处理结果

# - get\_result(): 获取处理结果

# - clear\_session(): 清理会话数据

# - export\_all\_results(): 导出所有结果

**2.4 页面文件设计**

**1\_📄\_Input\_Processing.py**

"""

文件路径: smart\_proposal\_engine/pages/1\_📄\_Input\_Processing.py

功能说明: 内容输入处理页面

"""

# 页面结构：

# 1. 输入方式选择（文件上传/文本输入）

# 2. 文件上传组件（支持音频、文档）

# 3. 文本输入区域

# 4. 处理选项配置

# 5. 处理进度显示

# 6. 结果预览和下载

# 复用组件：

# - 使用现有的文件上传UI组件

# - 复用进度显示逻辑

**2\_🔍\_Deep\_Analysis.py**

"""

文件路径: smart\_proposal\_engine/pages/2\_🔍\_Deep\_Analysis.py

功能说明: 深度分析页面

"""

# 页面结构：

# 1. 数据源选择（上传文件/使用上一步结果）

# 2. 分析模板选择下拉菜单

# 3. 自定义Prompt上传（可选）

# 4. 分析参数配置

# 5. 执行分析和进度显示

# 6. 分析报告预览和下载

**3\_📋\_Proposal\_Generation.py**

"""

文件路径: smart\_proposal\_engine/pages/3\_📋\_Proposal\_Generation.py

功能说明: 方案生成页面

"""

# 页面结构：

# 1. 分析报告上传/选择

# 2. 能力文档上传（可选）

# 3. 方案模板选择

# 4. 自定义模板上传（可选）

# 5. 生成参数配置

# 6. 方案预览和下载

**4\_🚀\_One\_Click\_Generation.py**

"""

文件路径: smart\_proposal\_engine/pages/4\_🚀\_One\_Click\_Generation.py

功能说明: 一键生成页面

"""

# 页面结构：

# 1. 批量文件上传区

# 2. 全流程参数配置

# 3. 执行进度显示（分步骤）

# 4. 结果汇总展示

# 5. 批量下载功能

# 特殊处理：

# - 复用各服务模块

# - 串联执行流程

# - 统一错误处理

**2.5 工具函数设计**

**file\_utils.py**

"""

文件路径: smart\_proposal\_engine/utils/file\_utils.py

功能说明: 文件处理相关工具函数

"""

# 主要功能：

# 1. 文件类型检测

# 2. 文件大小验证

# 3. 临时文件管理

# 4. 文件下载准备

# 5. 批量文件处理

# 复用现有代码：

# - 使用现有的format\_file\_size()

# - 使用现有的临时文件清理逻辑

**format\_utils.py**

"""

文件路径: smart\_proposal\_engine/utils/format\_utils.py

功能说明: 格式化相关工具函数

"""

# 主要功能：

# 1. 文本格式化

# 2. Markdown转换

# 3. 时间格式化

# 4. 数字格式化

# 复用现有代码：

# - format\_duration()

# - format\_metadata\_display()

**三、配置文件设计**

**3.1 app\_config.ini**

# 基于现有配置文件扩展

[API\_SETTINGS]

use\_internal\_api\_key = false

api\_key\_file = api\_key.txt

[MODEL\_SETTINGS]

transcription\_model = models/gemini-2.5-flash

analysis\_model = models/gemini-2.5-pro

proposal\_model = models/gemini-2.5-pro

[FEATURE\_SETTINGS]

enable\_deep\_analysis = true

enable\_proposal\_generation = true

enable\_custom\_prompts = true

enable\_batch\_processing = true

[TEMPLATE\_SETTINGS]

default\_analysis\_template = customer\_interview

default\_proposal\_template = project\_proposal

template\_directory = prompts/

[FILE\_SETTINGS]

max\_file\_size\_mb = 200

allowed\_audio\_formats = m4a,mp3,wav,aac,ogg,flac

allowed\_document\_formats = docx,pdf,txt

temp\_directory = temp/

output\_directory = output/

**3.2 requirements.txt**

# 基于现有依赖扩展

streamlit>=1.29.0

google-generativeai>=0.3.0

python-docx>=0.8.11

PyPDF2>=3.0.0

pydub>=0.25.1

pandas>=2.0.0

plotly>=5.0.0

python-dotenv>=1.0.0

**四、实施计划与优先级**

**4.1 第一阶段：基础框架搭建**

1. 创建项目目录结构
2. 设置基础配置文件
3. 实现BaseService基类
4. 迁移和适配现有服务代码

**4.2 第二阶段：核心功能实现**

1. 实现PromptManager
2. 完成各服务模块开发
3. 开发各功能页面
4. 实现数据流转机制

**4.3 第三阶段：集成与优化**

1. 实现一键生成功能
2. 添加错误处理
3. 优化用户体验
4. 编写使用文档

**五、与长期目标的兼容性保证**

**5.1 架构兼容性**

* 模块化设计便于后续扩展
* 统一接口便于集成新功能
* 配置驱动便于功能开关

**5.2 代码复用性**

* 最大程度复用现有代码
* 保持代码风格一致性
* 预留扩展接口

**5.3 数据兼容性**

* 统一的数据格式定义
* 标准化的元数据结构
* 向后兼容的配置设计

本实施方案确保了MVP版本的快速实现，同时为未来向完整版本演进奠定了坚实基础。通过模块化设计、统一接口和灵活配置，系统能够在保持简洁性的同时支持渐进式扩展。